On the existence of certain total recursive functions in nontrivial axiom systems, I.*

N. C. A. da Costa
F. A. Doria

Research Group on Logic and Foundations,
Institute for Advanced Studies, University of São Paulo.
Av. Prof. Luciano Gualberto, trav. J, 374.
05655–010 São Paulo SP Brazil.
NCACOSTA@USP.BR

Research Center on Mathematical Theories of Communication
and *Project Griffo*,
School of Communications,
Federal University at Rio de Janeiro.
Av. Pasteur, 250. 22295–900 Rio RJ Brazil.
DORIA@OMEGA.LNCC.BR

April 1998
Version 9.5

Abstract

We investigate the existence of a class of ZFC–provably total recursive unary functions, given certain constraints, and apply some of those results to show that, for Σ_1–sound set theory, ZFC $\nvdash P < NP$.

*Partially supported by CNPq, by FAPESP and by the PREVI Program, Federal University at Juiz de Fora.

Contents

1 Introduction **3**
Conventions, notations, Turing machines 4

2 Preliminary developments **4**
The machine V 6
The predicate $P^{\phi}(m, x)$ and the function $f_P^{\phi}(m)$ 7
Equivalence of representations 8

3 Nonexistence of a ZFC–provably total f_P^0 **8**

4 Application: results about the NP class **11**
The satisfiability problem 11
Polynomial machines 12
The NP class of problems 12
If ZFC is Σ_1–unsound then ZFC $\vdash P < NP$ 13
If ZFC is Σ_1–sound then ZFC $\nvdash P < NP$ 14
On the relativized result ZFC $\vdash P^A < NP^A$ 14
On the diagonalization procedure used here 15
$G?NG$ 16

5 Acknowledgments **16**

References **16**

1 Introduction

It is a classical fact that there are total unary recursive functions in the standard model for arithmetic which cannot be proved to be so within reasonably strong axiomatic systems (see [9], p. 257 and references therein). Such a phenomenon can especially be seen to happen within PA (Peano arithmetic) or within ZFC (Zermelo–Fraenkel set theory with the axiom of choice) if they satisfy a soundness condition, Σ_1–soundness (see Definition 2.1).

We consider in this paper a more particular situation: what can be said about a given partial recursive function when we add several specific restrictions to the objects considered within those Σ_1–sound systems? We obtain a partial unary recursive function which, if it is true that such a function is total, then that fact cannot be proved within those axiom systems, supposed consistent.

The original motivation for this paper stems out of the authors' search for incompleteness results with respect to nontrivial "rich" axiom systems, such as ZFC or a sizable fragment of it, which arise out of expressions for mathematical data, that is, which depend on the way we linguistically encode the mathematical objects. Some rather striking undecidable statements may arise in that way, and they settle noted questions as the integrability problem in classical mechanics, the decision problem for chaotic systems [3], or Arnol'd's problems on the nature of stability in dynamical systems [4, 5]. Those results not only settle open questions, but do so in a linguistic way, that is, they depend on the explicit form of expressions for mathematical concepts that may arise within a formal system. Moreover they are technically simple like the results presented here, and independent of the actual precise formal environment where they are placed.

We develop here the first part of a different (but also linguistically conceived) technique for the construction of independence results. The present technique comes down from a standard diagonal argument (see [10], p. 54, Ex. 5), and deals with provably recursive, denumerable objects which are kept so throughout the procedure. That technique is again rather insensitive to the actual formal setting where its proofs are supposed to happen, as we only require constructs from formal arithmetic in it.

We ask for the existence of a certain set of total recursive functions whose domain is informally understood as a set of expressions that describe Turing machines. We show here that it is consistent (with ZFC, taken as the formal background and supposed Σ_1–sound) to assume that those functions do not exist within the required axiom system.

As an application we discuss in the last Section the characterization problem for nondeterministic polynomial computations. We show how that question relates to the present work, as well as to previous work in the area. We then show that, if ZFC is Σ_1–sound, then ZFC $\nvdash P < NP$. Related and complementary results will be considered in Part II of this paper.

Conventions, notations, Turing machines

Remark 1.1 The formal setting is ZFC (Zermelo–Fraenkel set theory with the axiom of choice); informal definitions and proofs happen as usual within ordinary mathematics. Here we use Kleene's T predicate [8, 9]; ω is the set of natural numbers.

Restricted bounded arithmetic variables are in general used without making explicit their domains, since they are always clear by context. □

Remark 1.2 We require Turing machines in our argument; their behavior is specified below to avoid ambiguities:

1. Our Turing machines are defined over the set A_2^* of finite words on the binary alphabet $A_2 = \{0, 1\}$.
2. Each machine has $n + 1$ states $s_0, s_1, \ldots, s_n$, where s_0 is the final state. (The machine stops when it moves to s_0.)
3. The machine's head roams over a two–sided infinite tape.
4. Machines input a single binary word and either never stop or stop when the tape has a finite, and possibly empty set of binary words on it.
5. The machine's *output word* will be the one over which the head rests when s_0 is reached. (If the head lies on a blank square, then we take the output word to be the empty word.)

It is easy to formalize the concept of Turing machine within ZFC, as a Turing machine is a mathematical structure. For examples of such formalizations see [3]. In what follows we suppose that we are given the canonical enumeration of binary strings $\emptyset, 0, 1, 00, 01, 10, 11, \ldots$, and that each binary string is coded by the corresponding numeral in the sequence $0, 1, 2, 3, \ldots$. □

2 Preliminary developments

Definition 2.1 *A recursively axiomatizable theory R which is sufficient to develop elementary number theory is* Σ_1**–sound** *if and only if whenever $P(x)$ is a primitive recursive predicate such that $R \vdash \exists x\, P(x)$, there is a natural number n such that $P(n)$ holds.* □

Remark 2.2 We will make explicit when we suppose that either PA or ZFC are Σ_1–sound. □

Definition 2.3

1. *A* ZFC *unary function* f *is* ZFC–**provably total recursive** *if for some Gödel number* e_f *for* f, $\mathrm{ZFC} \vdash \forall x \in \omega \, \exists z \in \omega \, (T(e_f, x, z) \wedge \forall y \in \omega \, f(y) = \{e_f\}(y))$ [7].
2. *A* ZFC*–unary predicate* $Q(x)$ *is* ZFC–**provably total recursive** *if its characteristic function is* ZFC*–provably total recursive. Extension of that definition to a* n*–place predicate* $Q(x_0, x_1, \ldots, x_n)$ *is immediate.* □

We need a set $\mathcal{G}$ of ZFC–provably total recursive unary (t.r.u.) functions:

Definition 2.4 *The functions* $g_n \in \mathcal{G}$*, where* $n \in \omega$*, satisfy, for any* x *and any* m, n*:*

1. g_n *is* ZFC*–provably t.r.u.*
2. $g_0(x) > 2$.
3. $g_n(x+1) > g_n(x)$.
4. $g_n(x) > g_m(x)$, $n > m$. □

It is easy to see that infinitely many such sets $\mathcal{G}$ exist.

Let M_0, M_1, M_2, ..., be the single–tape Turing machines described above, and ordered by their Gödel numbers; we suppose that every $n \in \omega$ is a Gödel number for some machine.

Let $[g_n(|x|)]$, $n \in \omega$, $|x|$ the length of x, an input to M_i, be a clock that stops $\mathsf{M}_i(x)$ over input x after $g_n(|x|) - 1$ cycles. We suppose that the clock acts as follows: when the bound in the number of processing steps is reached, the clock stops the machine to which it is coupled and then moves its state to s_0. The machine's output is the word on which the head rests when the clock stops it. Let $\langle \ldots, \ldots \rangle$ denote the usual recursive 1–1 pairing function $\langle \ldots, \ldots \rangle : \omega \times \omega \to \omega$. We define:

Definition 2.5 *For* $p = \langle i, n \rangle$, $\mathsf{P}_p = \langle \mathsf{M}_i, [g_n] \rangle$. $\mathcal{P}^0 = \{\mathsf{P}_0, \mathsf{P}_1, \ldots\}$*; we order* $\mathcal{P}^0$ *by the code sequence* $p = 0, 1, 2, \ldots$. □

Remark 2.6 $\mathcal{P}^0$ contains all possible ordered pairs as in the preceding definitions; the number p can intuitively be seen as coding an expression for a Turing machine. From here onwards we will refer to those pairs when we talk about "expressions for Turing machines" of the kind considered in the present paper. □

Definition 2.7

1. *If there is a fixed $n \in \omega$ such that for every $x \in \omega$, $\mathsf{M}_i(x)$ stops before, or at most at $g_n(|x|)$ machine cycles, we say that M_i is $\mathcal{G}$–***bounded***.*
2. *A total recursive unary function $f(x)$ is $\mathcal{G}$–***bounded*** if there is a $\mathcal{G}$–bounded Turing machine that computes it.* □

Proposition 2.8 *If M_i is a $\mathcal{G}$–bounded machine, then there will be infinitely many $\mathsf{P}_p \in \mathcal{P}^0$ which compute the same function as M_i.* □

Remark 2.9 We can of course allow index n in g_n to range over an initial segment of the ordinals, given a notation system that satisfies the conditions spelled out in Definition 2.4. However we won't require this fact in the present paper. □

The machine V

Definition 2.10 *$\mathsf{V}(\langle x, s\rangle)$ is a fixed but otherwise arbitrary $\mathcal{G}$–bounded Turing machine such that:*

1. *V is an algorithm for a function $f_{\mathsf{V}} : \omega \times \omega \to \{0, 1\}$.*
2. *For every $x \in \omega$ there is an $s \in \omega$ such that $\mathsf{V}(\langle x, s\rangle) = 1$. Given the machine P_n, whenever $\mathsf{P}_n(x) = s$ and $\mathsf{V}(\langle x, s\rangle) = 1$, we say that x is* **acceptable for** *n.*
3. *For every $x \in \omega$ there is an $s \in \omega$ such that $\mathsf{V}(\langle x, s\rangle) = 0$.*
4. *For the empty word $\emptyset$ and for any nonempty strings x, s:*
   (a) $\mathsf{V}(\langle \emptyset, s\rangle) = 1$.
   (b) $\mathsf{V}(\langle x, \emptyset\rangle) = 0$.
   (c) $\mathsf{V}(\langle \emptyset, \emptyset\rangle) = 0$. □

Proposition 2.11 *Given a constant $x_0 \in \omega$, there are infinitely many P_m such that x_0 is acceptable for P_m.*

Proof: Let s be such that $\mathsf{V}(\langle x_0, s\rangle) = 1$. Let C_s be one of the constant Turing machines that input anything and output s. Let $c_s(|x_0|)$ be its operation time when input x_0 is fed to the machine. Then, given the conditions in Definition 2.4, it is easy to see that x_0 is acceptable for the $\mathcal{G}$–bounded machine

$\langle \mathsf{C}_s, g_{c_s(|x_0|)} \rangle$. And consequently, it is acceptable for all similar pairs, with clocks $[g_{(c_s(|x_0|)+k)}]$, for any natural k.

Moreover, since we have agreed that for the empty word $\emptyset$, we always have $\mathsf{V}(\langle x, \emptyset \rangle) = 0$, any x, one immediately sees that there are infinitely many $\mathcal{G}$–bounded Turing machines that, for a fixed x_0, will only accept that x_0, and no other input. □

The predicate $P^\phi(m,x)$ and the function $f_P^\phi(m)$

In what follows we use the coding of binary sequences by natural numbers. We will now slightly strengthen our notational conventions:

Definition 2.12 *A* **representation** *for $\mathcal{P}^0$ is any map $\phi : \mathcal{P}^0 \to \mathcal{P}^0$, such that:*

1. *$p \in \omega \mapsto \phi(p) \in \omega$ is a recursive permutation.*
2. *$\mathcal{P}^0 = \{\mathsf{P}_0^0, \mathsf{P}_1^0, \mathsf{P}_2^0, \ldots\}$, where $\mathsf{P}_0^0 = \mathsf{P}_0, \mathsf{P}_1^0 = \mathsf{P}_1, \ldots$*
3. *$\mathsf{P}_{\phi(p)}^\phi =_{\mathrm{Def}} \phi(\mathsf{P}_p^0)$.*
4. *$\phi\mathcal{P}^0 = \mathcal{P}^\phi = \{\mathsf{P}_0^\phi, \mathsf{P}_1^\phi, \ldots\}$.* □

Definition 2.13 *For $\mathsf{P}_m^\phi \in \mathcal{P}^\phi$, $P^\phi(m,x) \leftrightarrow_{\mathrm{Def}} \mathsf{V}(\langle x, \mathsf{P}_m^\phi(x) \rangle) = 0$.* □

Corollary 2.14 *$P^\phi(m,x)$ is a* ZFC*–provably recursive predicate.* □

Definition 2.15 *$f_P^\phi(m) =_{\mathrm{Def}} \mu_x P^\phi(m,x)$.* □

Corollary 2.16 *f_P^ϕ is partial recursive.* □

Remark 2.17 If ZFC is Σ_1–sound, then f_P^ϕ is also ZFC–provably partial recursive, as out of the ZFC–provably recursive characteristic function for P^ϕ we can compute a Gödel number for a machine that computes f_P^ϕ. □

One final result:

Corollary 2.18 $\mathrm{ZFC} \vdash \forall m \in \omega\, \exists y \in \omega\, P^\phi(m,y) \leftrightarrow f_P^\phi$ is total. □

Equivalence of representations

It doesn't matter which representation ϕ we choose:

Proposition 2.19 $\mathrm{ZFC} \vdash \forall m\, \exists x\, P^{\phi}(m,x) \leftrightarrow \forall n\, \exists x\, P^{\psi}(n,x)$.

Proof: $P^{\phi}(m,x) \leftrightarrow_{\mathrm{Def}} \mathsf{V}(x, \mathsf{P}^{\phi}_{m}(x)) = 0$. (We omit the $\langle \ldots \rangle$ for simplicity.) But $\mathsf{P}^{\phi}_{m}(x) = \mathsf{P}^{0}_{\phi^{-1}(m)}(x)$. (See Definition 2.12.) Then:

$$\mathsf{V}(x, \mathsf{P}^{\phi}_{m}(x)) = \mathsf{V}(x, \mathsf{P}^{0}_{\phi^{-1}(m)}(x)) = 0.$$

Thus

$$\mathrm{ZFC} \vdash P^{\phi}(m,x) \leftrightarrow P^{0}(\phi^{-1}(m), x),$$

and $\mathrm{ZFC} \vdash \exists x\, P^{\phi}(m,x) \leftrightarrow \exists x\, P^{0}(\phi^{-1}(m), x)$.

To deal with $\forall$ recall that ϕ is 1–1 and onto. Then $\mathrm{ZFC} \vdash \forall m \exists x\, P^{\phi}(m,x) \leftrightarrow \forall m \exists x P^{0}(\phi^{-1}(m), x)$, and

$$\mathrm{ZFC} \vdash \forall m\, \exists x\, P^{\phi}(m,x) \leftrightarrow \forall n\, \exists x\, P^{0}(n,x)$$

(with $n = \phi^{-1}(m)$), whereas the conclusion, since the argument holds for arbitrary ϕ. □

Corollary 2.20 $\mathrm{ZFC} \vdash \forall \phi\, [\phi$ is a representation $\rightarrow (\forall m\, \exists y\, P^{\phi}(m,y) \leftrightarrow f^{\phi}_{P}$ is total)]. □

3 Nonexistence of a ZFC–provably total f^{0}_{P}

In what follows we will frequently refer to "machines" for the sake of a more intuitive argument.

Proposition 3.1 *There is a representation ϕ such that, if* ZFC *is Σ_1–sound, for every* ZFC*–provably total recursive unary (t.r.u.) function F_i,* $\mathrm{ZFC} \vdash f^{\phi}_{P} \neq F_i$.

Remark 3.2 Therefore either f^{ϕ}_{P} is partial or it is total but not ZFC–provably total.

The idea of the proof is the following: we will recursively reorder the sequence $\mathcal{P}^0$ to obtain $\mathcal{P}^{\phi}$ which is such that every ZFC–provably t.r.u. function will differ somewhere from f^{ϕ}_{P} (see Definition 2.15). More precisely:

- Recall that $P^{\phi}(m,x) \leftrightarrow \mathsf{V}(\langle x, \mathsf{P}^{\phi}_{m}(x)\rangle) = 0$ and that $f^{\phi}_{P}(m) = \mu_x P^{\phi}(m,x)$, where m is the machine code in $\mathcal{P}^{\phi}$ and x is the machine's input.
- If $y = \mu_x P^{\phi}(m,x) = f^{\phi}_{P}(m)$, then $\mathsf{V}(\langle y, \mathsf{P}^{\phi}_{m}(y)\rangle) = 0$, or equivalently by definition, $P^{\phi}(m,y)$ holds.

- Then if $y' = F_i(m)$, and if $\neg P^\phi(m, y')$ is provable, then that last statement is equivalent to $\mathsf{V}(\langle y', \mathsf{P}^\phi_m(y')\rangle) = 1$.
- Therefore $y' \neq y = f^\phi_P(m) = \mu_x P^\phi(m, x)$.
- Therefore $y' = F_i(m) \neq f^\phi_P(m)$.

We must arrange $\mathcal{P}^\phi$ so that for any i and for at least one m, $y' \neq y$. □

Proof: We will consider two sequences of functions and machines: the first one, kept fixed throughout the proof, is the sequence $F_0, F_1, \ldots$, of ZFC–provably t.r.u. functions. The second sequence is the sequence $\mathsf{P}^0_0, \mathsf{P}^0_1, \ldots$ of $\mathcal{G}$–machines, which will eventually be moved to and fro during the proof.

Start theorem–proving in ZFC. Pick up all theorems of the form

$$\text{ZFC} \vdash \forall x \in \omega\, \exists z \in \omega\, T(e, x, z) \wedge \forall y \in \omega\, (f(y) = \{e\}(y)). \tag{1}$$

(See Lemma 4.10.) We thus generate a r.e. sequence of Gödel numbers

$$e_{f_0} = e_0, e_{f_1} = e_1, e_{f_2} = e_2, \ldots.$$

Repetitions are allowed—just pick up the Gödel numbers as they come up in the sequence. Out of them obtain the sequence of Turing machines

$$\mathsf{M}_{e_0}, \mathsf{M}_{e_1}, \ldots$$

which compute the ZFC–provably recursive total functions

$$\{e_0\}(x), \{e_1\}(x), \ldots$$

We use the notation: F_j is computed by M_{e_j}. We then reconstruct the sequence F_j (out of the indices e_j) within ZFC, where the rest of the argument happens.

Within ZFC, let $\mathcal{P}^0$ be given, ordered by the indices p of the P_p. (See Definitions 2.5 and 2.12.)

- *Step* 0: We obtain a f^ϕ_P such that $F_0 \neq f^\phi_P$. The idea is to reshuffle $\mathcal{P}^0$ to suit our purposes. First, generate the list of theorems of ZFC up to

  $$\text{ZFC} \vdash \forall x \in \omega\, \exists z \in \omega\, T(e_0, x, z) \wedge \forall y \in \omega\, (f(y) = \{e_0\}(y)).$$

  Within ZFC, F_0 is computed by M_{e_0}. Get $\mathsf{M}_{e_0}(0) = y'_0$. Start from $\mathcal{P}^0$. Call $f^\phi_P(0)$ (computed out of $\mathcal{P}^0$) the "provisional" $f^\phi_P(0)$. Now for the provisional value, either $f^\phi_P(0)$ doesn't exist, or $f^\phi_P(0) = y_0$.

  - If the provisional $f^\phi_P(0)$ doesn't exist, then f^ϕ_P is a partial function, and the proof stops here. Then the provisional value is the definitive one.

– If, for the provisional value, $f_P^\phi(0) = y_0$, we will ensure that $y_0 \neq y_0'$. If required, we will change things (see below) so that for the definitive value y_0 we ensure that $y_0 \neq y_0'$.

There are again two possibilities here:

– If $\mathsf{V}(\langle y_0', \mathsf{P}_0^0(y_0')\rangle) = 1$, proceed to the next step, and put $\mathsf{P}_0^\phi = \mathsf{P}_0^0$, the rest being left as in $\mathcal{P}^0$ until the next step is reached.

In that case go to an $m = m_1 = k_0 + 1 > 0$ (see below) and execute the next step.

– If $\mathsf{V}(\langle y_0', \mathsf{P}_0^0(y_0')\rangle) = 0$, let $\mathsf{P}_{k_0}^0$ be the first $\mathcal{G}$–bounded machine in $\mathcal{P}^0$ such that

$$\mathsf{V}(\langle y_0', \mathsf{P}_{k_0}^0(y_0')\rangle) = 1.$$

(In order to make that test compute $\mathsf{V}(\langle y_0', \mathsf{P}_j^0(y_0')\rangle)$, $j = 0, 1, 2, \ldots$, up to the first index $j = k_0$ such that $\mathsf{V}\ldots = 1$. Recall that we deal with total machines and that there are infinitely many machines so that y_0' is acceptable; pick a machine like the one last described in Proposition 2.11.)

If that machine accepts all instances, then the function is partial and the proof stops here. If not, it will reject some instance y, and since y' is accepted, $y' \neq y$.

Then change places between that machine $\mathsf{P}_{k_0}^0$, and P_0^0, so that we obtain an initial segment of $\mathcal{P}^\phi$, $\mathcal{P}_0^\phi$ given by $\mathsf{P}_{k_0}^0, \mathsf{P}_1^0, \mathsf{P}_2^0, \ldots, \mathsf{P}_{k_0-1}^0, \mathsf{P}_0$. (That sequence is relabeled:

$$\mathsf{P}_0^\phi = \mathsf{P}_{k_0}^0, \mathsf{P}_1^\phi = \mathsf{P}_1^0, \mathsf{P}_2^\phi = \mathsf{P}_2^0, \ldots, \mathsf{P}_{k_0-1}^0, \mathsf{P}_{k_0}^\phi = \mathsf{P}_0^0$$

and gives the initial segment $\mathcal{P}_0^\phi$ of $\mathcal{P}^\phi$.) Clearly here $F_0(0) \neq f_P^\phi(0)$, since f_P^ϕ either diverges at 0 or differs there from F_0. The definitive value of $f_P^\phi(0)$ is also determined here out of the rearranged sequence of the Ps.)

Leave step 0.

- *Step* 1 : Out of the listing of theorems of ZFC obtain

$$\mathrm{ZFC} \vdash \forall x \in \omega\, \exists z \in \omega\, T(e_1, x, z) \wedge \forall y \in \omega\, (f(y) = \{e_1\}(y)).$$

We now obtain $F_1 \neq f_P^\phi$. Situation is the same as in the first step, with 1 substituted throughout for 0 in the lower indices. For the nontrivial case in Step) we get: $\mathcal{P}_0^\phi \cup \{\mathsf{P}_{k_1}^0, \mathsf{P}_{k_0+2}^0, \ldots, \mathsf{P}_{k_1-1}^0, \mathsf{P}_{k_0+1}^0\}$ (with a slight abuse of notation), which is the extension $\mathcal{P}_1^\phi$ of $\mathcal{P}_0^\phi$ achieved at the present step.

- ...

- *Step* n: Obtain

$$\text{ZFC} \vdash \forall x \in \omega \, \exists z \in \omega \, T(e_n, x, z) \wedge \forall y \in \omega \, (f(y) = \{e_n\}(y)).$$

We get $F_n \neq f_P^\phi$. .... Same procedure. □

Claim 3.3 *$\mathcal{P}^\phi$, the reordered sequence of $\mathcal{G}$–bounded machines that results from the extension of all initial segments $\mathcal{P}_m^\phi$, is a recursive sequence of $\mathcal{G}$–bounded machines.*

Proof: The map $\phi : \mathcal{P}^0 \to \mathcal{P}^0$ is clearly 1–1 and total: let us be given $\mathsf{P}_n \in \mathcal{P}$. To compute $\phi(\mathsf{P}_n)$, obtain an ordered initial segment $\mathcal{P}_j^\phi$ as above of cardinality $> n+1$; $\phi(\mathsf{P}_n)$ is the machine at position n in that segment $\mathcal{P}_m^\phi$. The converse operation is immediate, given $\mathcal{P}^\phi$, out of the definition of $\mathcal{P}^0$. □

Claim 3.4 *For each i,* $\text{ZFC} \vdash f_P^\phi \neq F_i$. □

Claim 3.5 *For each i,* $\text{ZFC} \vdash \exists x \, [\neg P^\phi(x, F_i(x))]$. □

Corollary 3.6 ZFC *is Σ_1–sound (see Definition 2.1) if and only if* ZFC $\nvdash f_P^\phi$ is total.

Proof: We have shown that a proof of "f_P^ϕ is total" cannot appear in the listing of all proofs of ZFC. □

¿From the preceding claims and from Proposition 2.19 and Corollaries 2.20 and 3.6:

Corollary 3.7 *If* ZFC *is Σ_1–sound then* ZFC plus [The partial recursive function f_P^0 isn't total] *is consistent.* □

4 Application: results about the NP class

We consider here the NP class of problems.

The satisfiability problem

Remark 4.1 The motivation here comes from the satisfiability problem for Boolean expressions in conjunctive normal form (cnf).

Suppose that we write a predicate $A^\phi(m, x)$ which means "polynomial machine P_m^ϕ accepts x," that is, P_m^ϕ inputs a binary string x which polynomially codes a Boolean expression in cnf and outputs another binary string s which satisfies that same instance x:

- We can in fact polynomially encode each Boolean expression in cnf as a binary string; that coding may be constructed as an onto map Sat $\rightarrow \omega$, where Sat is the set of all such Boolean expressions (tautologies and totally false expressions excluded).
- If $x \in$ Sat, adequately encoded, and if P_n is the n–th polynomial machine in the listing given above, let us write $s = \mathsf{P}_n(x)$.
- V is the "verifying machine" that polynomially checks whether s satisfies x or not. That is the meaning of $\mathsf{V}(x, s) = 1$ (s satisfies x) or $\mathsf{V}(x, s) = 0$ (s doesn't satisfy x).

The conditions we impose on V (see Definition 2.10) arise out of that motivation. □

Polynomial machines

Remark 4.2 Let the set $\mathcal{G}$ (Definition 2.4) be the set of polynomials:

1. $p_0(x) = x^3 + 3$.
2. $p_n(x) = x^{n+3} + (n + 3)$. □

Proposition 4.3 *The set of $\mathcal{G}$–bounded Turing machines whose bounds are given by Remark 4.2 is (essentially) the set of all polynomial machines.*

Proof: Every polynomial machine can be represented as a pair $\langle \mathsf{M}_n, [p_k] \rangle$. And every Turing machine coupled to a polynomial clock such as $\langle \mathsf{M}_n, [p_k] \rangle$ is polynomial. □

(See [1].) Throughout this Section, $\mathcal{P}^0$ is a set of expressions that represent polynomial Turing machines. Let $\mathsf{V}(x, s)$ be the (fixed) polynomial machine that checks whether s solves instance x (Definition 2.10). If $\mathsf{V}(x, s) = 1$, we say that s satisfies x.

The NP class of problems

Next definition characterizes the NP class [1, 12]:

Definition 4.4 *Let x, y be binary words (identified to the corresponding numbers), let p be a polynomial, and let $R(x, y)$ be a 2–place recursive polynomial predicate. Then:*

1. $x \in A_{p,R} \leftrightarrow \exists y \, [|y| \leq p(|x|) \wedge R(x, y)]$.
2. $NP = \{A_{p,R} : p$ is a polynomial and R is a 2–place recursive polynomial predicate$\}$. □

Then, after Definition 4.4:

Definition 4.5 $x \in \mathrm{SAT} \leftrightarrow \exists s\,[|s| \leq p_*(|x|) \wedge \mathsf{P}(x) = s \wedge \mathsf{V}(\langle x, s\rangle) = 1]$, *where* p_* *is a fixed polynomial, and* P *is a polynomial machine.* □

Remark 4.6 The condition $p_*(|x|) \geq |s|$ directly follows from the satisfiability problem. □

Definition 4.7 *For* $\mathsf{P}^\phi_m \in \mathcal{P}^\phi$,

$$A^\phi(m,x) \leftrightarrow_{\mathrm{Def}} [\mathsf{V}(\langle x, \mathsf{P}^\phi_m(x)\rangle) = 1] \leftrightarrow \exists s\,[\mathsf{P}^\phi_m(x) = s \wedge \mathsf{V}(\langle x, s\rangle) = 1].\square$$

Definition 4.8 (Formalization of $P < NP$ for SAT.) *The $P < NP$ conjecture (for* SAT*) in* ZFC *is:*

$$\forall m \in \omega\, \exists x \in \omega\, \neg A^0(m,x).\square$$

Proposition 4.9 *If* $f^0_{\neg A}(m) = \mu_x \neg A^0(m,x)$, *then*

$$\mathrm{ZFC} \vdash (\forall m \in \omega\, \exists x \in \omega\, \neg A^0(m,x)) \leftrightarrow [f^0_{\neg A} \text{ is total}].\square$$

If ZFC is Σ_1–unsound then ZFC $\vdash P < NP$

Lemma 4.10 *Let* R *be a recursively axiomatizable simply consistent extension of* PA *which is not* Σ_1*–sound. Then every partial recursive function is equal to a function which is provably total from* R. *That is, for each* e, *there is an* a *such that* $\varphi_e = \varphi_a$, *and* $R \vdash [\varphi_a \text{ is total}]$.

Proof: To construct a, fix a primitive recursive predicate $Q(x)$ such that $R \vdash \exists x\, Q(x)$, but such that $Q(n)$ is false for each natural number n. (Possible, since R isn't Σ_1–sound.) In Kleene Normal Form [8], we have $\varphi_e(x) = U(\mu_y T(e,x,y))$, where U, T are as in the reference; recall that they are primitive recursive. Then define $\varphi_a(x) = U[\mu_y(T(e,x,y) \vee Q(y))]$. $\varphi_a = \varphi_e$, since $Q(y)$ is always false, but $R \vdash \forall x\, \exists y\, [T(e,x,y) \vee\ Q(y)]$, so $R \vdash [\varphi_a \text{ is total}]$. □

Remark 4.11 In order to obtain one such R, for example pick up PA and if P is primitive recursive and $\forall x\, \neg Q(x)$ is true while independent of PA, take $R = \mathrm{PA} + \exists x\, Q(x)$. □

Corollary 4.12 *If* ZFC *isn't* Σ_1*–sound, then* $\mathrm{ZFC} \vdash P < NP$.

Proof: From Proposition 4.9 and Lemma 4.10. □

If ZFC is Σ_1–sound then ZFC $\nvdash P < NP$

Proposition 4.13 *If* ZFC *is* Σ_1*–sound, then:*

1. $f^0_{\neg A}$ *isn't* ZFC*–provably total.*
2. ZFC $\nvdash P < NP$*.*
3. ZFC *is consistent if and only if* ZFC $+ (P = NP)$ *is consistent.*

Proof: From Proposition 4.9 and Corollary 3.7. □

Corollary 4.14 *If* ZFC *is consistent, then* ZFC *is* Σ_1*–sound if and only if* ZFC $\nvdash P < NP$*.* □

Remark 4.15 We've just added Corollary 4.14 as a special assertion in order to emphasize the depth of the $P?NP$ question, which leads to such a beautiful interplay between formal systems and the surrounding metamathematics. The previous results are certainly valid for any Σ_1–sound fragment of ZFC which contains PA. □

On the relativized result ZFC $\vdash P^A < NP^A$

Remark 4.16 We comment here on the well–known results in [1]. Some preliminary remarks must be made before we get into details:

- We first notice that the Turing machines considered in the present paper paper are of a more restricted kind than those in the reference, as (in our case) the only special state is s_0 (see Remark 1.2). Actually our special machine V stands for their "accepting states," with several restrictions added (see Definition 2.10).

  The restrictions we impose on V are of course modelled on the way one mechanically verifies the validity of a choice of truth values for Boolean expressions in cnf, and on their possible choices.

- We also recall that, for oracle machines where the oracle is realized as an extra tape in a multitape Turing machine with the oracle set written on it, if $P = NP$ then $P^A = NP^A$, for any oracle A ([1], Remark in p. 437).

- Yet in [1] one supposes that the oracle is consulted, and gives its answer, in a single step.

But the point is, in which way (if any) does our construction in Proposition 3.1 conflict with, say, the one in the same Theorem 3?

- First of all, in their proof of Theorem 3 they use the oracle in an essential way, that is, all machines must *nontrivially* refer to the oracle during their computations. That is to say, we exclude situations like the machine with the oracle tape which is never accessed by its program, or the machine that proceeds in the same direction never caring whether it does get a "yes" or "no" from the oracle.

  So, from start we notice that the freedom we enjoyed in moving machines to and fro in the proof of Proposition 3.1 may not be available in the (very restrictive) relativized setting, as we proceed with the constructions in the reference.

- But let us consider the proof of Theorem 3 in the reference [1]: in Theorem 3 one aims at the construction of a language $L(B)$ which will be rejected by every polynomial oracle machine P_i^B which nontrivially consults with oracle B. That is to say, at least one instance $x \in L(B)$ isn't accepted by each P_i^B.

- If B is the oracle, then $L(B) = \{y : |y| = |x| \wedge x \in B\}$.

- The proof in the reference proceeds by a stepwise construction of B; if step i precedes step j, then $B_i \subseteq B_j$, and $B = \bigcup_i B_i$. Moreover, B can be made recursive.

- Once one has $L(B)$, we can obtain a provably total recursive function $h_B(i) = \min(y : |y| = |x_i| \wedge x_i \in B)$.

- It is clear that $h_B(i) \neq f_{\neg A}^B(i)$.

The essential point is: it is not at all clear whether we would be able to make the permutations which are essential to the proof of Proposition 3.1 in the present restricted, relativized setting. □

On the diagonalization procedure used here

Remark 4.17 We must also add a few comments on the diagonalization used here. Diagonal constructions have been used out of a 'lower' class X to obtain objects in a 'higher' class $Y \supset X$. More precisely, out of some listing of classes of NP–languages which are known to be in P, we try to obtain via a diagonal construction a language which isn't in P.

Our procedure in this paper elaborates on a well–known construction (see [10], p. 54, Ex. 5). We temporarily forget about P and NP and just consider the relation between f_P^0 and the listing of all ZFC–provably total unary recursive functions. We also try to impose some properties on that f_P^0, such that it can be proved to be Turing–computable, and one can prove that there is a Gödel number for the corresponding machine.

The diagonal construction is then used to show that such a function differs from each one of the ZFC–provable t.r.u. functions at least once. To put it in another way, instead of using a single sequence of functions to diagonalize out of it, we couple two sequences, the fundamental one—the sequence of the F_i—and the subordinate sequence, the P_j. □

G?NG

Remark 4.18 M. Benda has pointed out to the authors that their original construction leads to a whole hierarchy of problems of the form $G?NG$, which is why we have decided to frame our results in that wider setting from start [2]. □

5 Acknowledgments

Notice that both Definition 2.1 and Lemma 4.10 are crucial to this work. On the proof of Proposition 3.1 see [10].

Portions of this work were written while the second author was on leave at the Graduate Studies Program in Communications at the Federal University at Piauí and later at FACOM, Federal University at Juiz de Fora (Brazil). Their hospitality is thankfully acknowledged. Formatting of this paper is due to *Project Griffo* at the Rio de Janeiro Federal University; thanks are due to E. Carneiro Leão, Muniz Sodré, R. Paiva, J. Argolo and P. Pires.

Internet facilities for FAD were provided by the M. Kritz Research Program at the LNCC–CNPq; its support and help is gratefully acknowledged.

Finally both authors wish to thank CNPq, CAPES, and FAPESP, Philosophy Section, for support of the present work.

References

[1] T. Baker, J. Gill, R. Solovay, "Relativizations of the $P =?NP$ question," *SIAM Journal of Computing* **4**, 431 (1975).

[2] M. Benda, e–mail message to the authors (1997).

[3] N. C. A. da Costa and F. A. Doria, "Undecidability and incompleteness in classical mechanics," *International Journal of Theoretical Physics* **30**, 1041 (1991).

[4] N. C. A. da Costa and F. A. Doria, "An undecidable Hopf bifurcation with an undecidable fixed point," *International Journal of Theoretical Physics* **33**, 1913 (1994).

[5] N. C. A. da Costa and F. A. Doria, "Gödel incompleteness in analysis, with an application to the forecasting problem in the social sciences," *Philosophia Naturalis* **31**, 1 (1994).

[6] R. Dougherty and T. Jech, "Left–distributive embedding algebras," *Electronic Research Anouncements, American Mathematical Society* **3**, 28 (1997).

[7] J. Ketonen, R. Solovay, "Rapidly rising Ramsey functions," *Annals of Mathematics* **113**, 267 (1981).

[8] S. C. Kleene, *Introduction to Metamathematics*, Van Nostrand (1952).

[9] S. C. Kleene, *Mathematical Logic*, John Wiley (1967).

[10] K. Kunen, *Set Theory*, North–Holland (1983).

[11] C. Smorynski, "The incompleteness theorems," in J. Barwise, ed., *Handbook of Mathematical Logic*, North–Holland (1989).

[12] L. Stockmeyer, "Classifying the computational complexity of problems," *Journal of Symbolic Logic* **52**, 1 (1987).